(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 December 2000 (21.12.2000)

PCT

(10) International Publication Number
WO 00/77684 A1

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US00/14072

(22) International Filing Date: 22 May 2000 (22.05.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/139,012 14 June 1999 (14.06.1999) US

(71) Applicant: VIRTUALSTEP, INC. [US/US]; 45 Amador Village Circle #37, Hayward, CA 94544 (US).

(72) Inventor: CHIOU, Yee, Sue; 45 Amador Village Circle #37, Hayward, CA 94544 (US).

(74) Agents: GALLIANI, William, S. et al.; Pennie & Edmonds LLP, 1155 Avenue of the Americas, New York, NY 10036 (US).

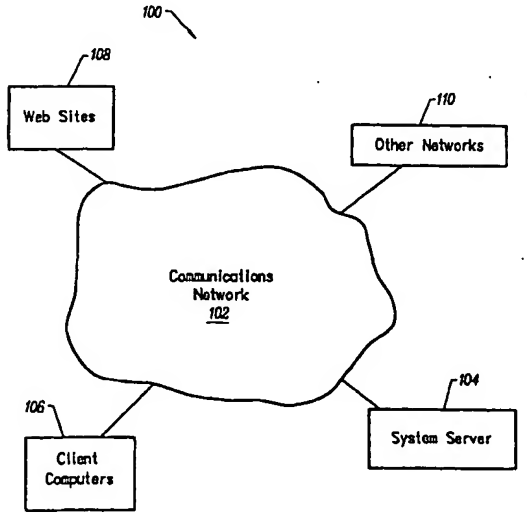
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: APPARATUS AND METHODS FOR DEVELOPING AND MANAGING SOFTWARE INTER-OPERABILITY AND DATA INTEGRATION ACROSS INFORMATION SYSTEMS



(57) Abstract: The present invention provides apparatus and methods to achieve software inter-operability and data integration among information systems (100). Project data is retrieved from different software and encapsulated by developed objects. Developed objects are grouped into object groups. The encapsulated data is accessible from a centralized data repository (104) across a network (102). When requested by a client computer (106), encapsulated data is converted into network compatible format, a report is generated from the data and sent to the client computer (106).

WO 00/77684 A1

APPARATUS AND METHODS FOR DEVELOPING AND MANAGING SOFTWARE INTER-OPERABILITY AND DATA INTEGRATION ACROSS INFORMATION SYSTEMS

This application claims priority of the provisional application bearing application number 60/139,012 filed on June 14, 1999.

BRIEF DESCRIPTION OF THE INVENTION

This invention relates to software inter-operability. In particular, this invention relates to software inter-operability and data integration among information systems.

BACKGROUND OF THE INVENTION

The management of large-scale projects, such as construction projects, requires collaboration among and between a wide array of individuals and software systems. Typically, different software systems are each targeted to manage data in different stages of a project. For example, in the context of construction projects, various software programs are used to assist planning and implementation at different stages of a project. Commonly used software programs in the context of managing construction projects include, but are not limited to, the following:

1. Computer-aided Design Systems (CAD) that define the geometrical shapes and design parameters of physical entities; examples of CAD systems are Autodesk AutoCAD® and Autodesk Architecture Desktop®, Autodesk, San Rafael, CA, and Bentley Microstation®, Bentley System, Exton, PA;

2. Scheduling and Project Management Systems that provide project scheduling, project control and resource management; examples of scheduling and project management systems are Primavera Project Planner®, Primavera System, Bala Cynwyd, PA, and Microsoft Project®, Microsoft, Redmond, WA;

3. Cost Estimating Systems that provide cost estimation for a project; examples of cost estimating systems are Timberline Precision Collection® and Timberline Gold Collection®, Timberline, Beaverton, OR;

4. Graphical Reporting Systems that publish a variety of project reports in graphical chart formats; examples of graphical reporting systems are Microsoft Excel®, Microsoft, Redmond, WA, and Lotus 1-2-3®, Lotus, Cambridge, MA; and

5. Database Management Systems that maintain the persistence of software instances; examples of database management systems are Oracle® Database Server, Oracle, Redwood Shores, CA, Microsoft SQL Server® and Microsoft Access®, Microsoft, Redmond, WA.

Data manageable by a software program is generally limited to the capabilities of that software program. Data exchange among different software programs is limited and usually inflexible. Thus, it is desirable to provide apparatus and methods which are capable of extracting interrelated project data residing in different software applications and incorporate such interrelated data in a centralized database for each project. Further, it is desirable to provide access to such centralized database through a network, such as the Internet.

SUMMARY OF THE INVENTION

A method for processing data received across a network in multiple formats includes the steps of: (a) receiving data at a server computer from a client computer; (b) identifying a selected parser program based at least in part on the received data; (c) parsing the received data using the selected parser program to create parsed data; (d) selectively establishing assigned portions of the parsed data to a target object in a database; (e) encapsulating the assigned portions within the target object to form a new target object; and (f) storing the new target object in the database. In an exemplary embodiment, the method also includes the steps of: (g) receiving a request from the

client computer; (h) selectively retrieving data stored in the database in response to the request to form retrieved data; (i) converting the retrieved data into a network compatible format to form converted data; (j) generating a report based on the converted data; and (k) sending the report to the client computer.

In one embodiment, the converting step includes the step of forming converted data in a HTML or XML format. In another embodiment, the identifying step includes the step of identifying a selected parser program from a set of parser programs. In yet another embodiment, the method includes the step of inter-relating a set of target objects in a relational table structure.

In an exemplary embodiment, a method for processing data received across a network in multiple formats includes the steps of: (a) uploading a document provided by a client computer; (b) identifying a selected parser program based at least in part on the document; (c) parsing the document using the selected parser program to provide a parsed document; (d) identifying an entity from the parsed document; and (e) associating the entity to an object in a schedule. In one embodiment, the uploading step includes uploading a CAD document.

A system for processing data received across a network in multiple formats includes a central processing unit, a communication interface for connection between the network and the central processing unit, and a memory. In one embodiment, the memory includes a set of data parser programs and a database having a set of objects. Data received through the communication interface is processed by the central processing unit utilizing at least one of the set of data parser programs to create processed data, the processed data being selectively associated with one of the set of objects in the database and stored in the database based on the association. In another embodiment, the memory further comprises a report generator module. The report generator module includes logic code to selectively produce retrieved data from the database in response to a request from a client computer across the network, logic code to convert the retrieved data into converted data in a network compatible format, and logic code to send the converted data across the network to the client computer. In one embodiment, the network is a distributed network and the client computer includes an end user or a web site.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 illustrates an exemplary system in accordance with an embodiment of the invention.

FIGURE 2A illustrates an exemplary system server in accordance with an embodiment of the invention.

FIGURE 2B illustrates an exemplary system in accordance with an embodiment of the invention.

FIGURE 3 is a flow chart of an exemplary process in accordance with an embodiment of the invention.

FIGURE 4A is a flow chart of an exemplary process in accordance with an embodiment of the invention.

FIGURE 4B is a flow chart of an exemplary process in accordance with an embodiment of the invention.

FIGURE 4C is a flow chart of an exemplary process in accordance with an embodiment of the invention.

FIGURE 5 illustrates an exemplary inter-relationship of objects in an objects database.

DETAILED DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a network system 100 in accordance with an embodiment of the present invention. The network system 100 includes a communications network 102 connected to a system server 104, client computers 106, Web sites 108, and other networks 114. In a preferred embodiment, the communications network 102 is the Internet.

Figure 2A is an exemplary embodiment of the system server 104 in accordance with an embodiment of the present invention. The system server 104 includes one or more central processing units 202, a communications interface 204 for connecting to the communications network 102, and memory 206. The memory 206 includes an operating system 208, server applications 210, communications applications 212, parser programs for parsing incoming documents received through the communications network 102, a report generator 216 for generating reports, a main

database 218, a format converter 234 for converting formats, and a relationship compiler 236 for compiling data and object inter-relationships.

The main database 218 includes an objects database 220 for storing objects, a programs database 222 for storing computer programs, and a systems application program interfaces 232. The programs database 222 includes scheduling programs 224, cost estimating programs 226, graphical report programs 228, and computer aided design (CAD) programs 230.

Figure 2B is an exemplary embodiment of the communication between the system server 104 and the client computers 106 through the communications network 102. In this exemplary embodiment, client computers 106 can upload native data through the communications network 102 for processing by the system server 104. The client computers's native data includes drawings 201, schedules 203, and financial files 205. Once received by the system server 104, the data is parsed by a parser program 214. The parser program 214 is selected from a set of parser programs. After the data is parsed, inter-relationships among the parsed data and objects in the objects database 220 are developed by the relationship compiler 236. The parsed data is stored in the main database 218 in accordance with the parsed data's inter-relationships to objects in the objects database 220 of the main database 218. Data associated with each object is encapsulated by that object. New objects may also be developed to accommodate the parsed data.

Data stored in the main database 218 can be accessed by client computers 106 or Web sites 108 through the communications network 102. When the system server 104 receives a request from the client computers 106 or Web sites 108 through the communication network 102, data responsive to the request is retrieved from the main database 218. The report generator 216 generates a report based on the retrieved data. The generated report is then converted into a network compatible format by the format converter 234 and sent to the requesting client computers 106 or Web sites 108.

Figure 3 illustrates an exemplary process in accordance with an embodiment of the present invention. At step 302, data is received from a client. Depending on the content of the data received, a parser program is selected (step 304). The received data is parsed by the selected parsing program (step 306). Inter-relationships among the parsed data and objects in an objects database are developed (step 308). The parsed

data is stored in a main database, organized by the parsed data's inter-relationships with objects in the objects database (step 309). New objects may also be developed to accommodate the parsed data. Upon request from a client across a network, such as the Internet, responsive data is retrieved from the main database and a report is generated from retrieved data (step 310). The generated report is converted into a network compatible format (step 312). The converted report is sent across the network to the requesting client (step 314). The client can either be a client computer accessed by a end user or a Web site.

Figure 4A illustrates an exemplary process in accordance with an embodiment of the present invention. At step 402, a user is allowed to sign on to the server system. After the user is signed on, a new project is initialized (step 404). The user is prompted to define work areas (step 406). In an exemplary embodiment, the defined work areas form work area objects. Examples of work areas are various floors in a construction project. The user is prompted to select building blocks for each work area (step 408). In an exemplary embodiment, the selected building blocks form building block objects. Examples of building blocks are number of columns, walls, girders, etc. Building elements and actions are developed based on the defined work areas and selected building blocks (step 410). In an exemplary embodiment, the developed building elements and actions form building element objects and action objects, respectively. Building elements are developed by associating building blocks to each work area, for example, two columns on the first floor. Actions are developed by defining the work necessary to achieve each building element, for example, welding the columns onto the first floor. Activities are developed based on the developed building elements and actions (step 412). In an exemplary embodiment, the developed activities form activity objects. Activities are developed based on when each action should be accomplished, for example, welding columns onto the first floor on the fifteenth day of the construction project. A schedule listing detail activities on a day-to-day basis is generated (step 414). At step 415, inter-relationships among the objects are developed and each object is stored in a main database in accordance with the developed inter-relationships.

In an exemplary embodiment, the system server 104 is able to service clients using different computer software to generate documents. For example, using the

context of a construction project, a manager of the construction project may use cost estimating software to generate estimated cost and an architect of the same construction project may use computer-aided design software to sketch the structure of the construction. In each case, the system server 104 of this invention can upload the documents created by individuals in a project using different computer software, compile the data in such uploaded documents, generate a centralized database, and provide a comprehensive report to each individual having access to the system regarding the construction project.

Figure 4B illustrates an exemplary process in accordance with an embodiment of the present invention. In this exemplary process, the user has at least one document already created using a computer-aided design software. At step 416, the user is allowed to sign on to the server system. After the user is signed on, a new project is initialized (step 418). The user is prompted to define work areas (step 420). In an exemplary embodiment, the defined work areas form work area objects. The user's CAD drawings are uploaded into the server system (step 422). Entities from the CAD drawings, or CAD entities, are identified (step 424). Building elements and actions are developed based on the identified entities (step 426). In an exemplary embodiment, the developed building elements and actions form building element objects and action objects, respectively. Activities are developed based on the developed building elements and actions (step 428). In an exemplary embodiment, the developed activities form activity objects. A schedule listing detail activities on a day-to-day basis is generated (step 430). At step 431, inter-relationships among the objects are developed and each object is stored in a main database in accordance with the developed inter-relationships.

Figure 4C illustrates an exemplary process in accordance with an embodiment of the present invention. In this exemplary process, the user has more than one document already created using computer-aided design software and scheduling software. At step 432, the user is allowed to sign on to the server system. After the user is signed on, a new project is initialized (step 434). The user is prompted to define work areas (step 436). The user's documents, including a CAD drawing and a schedule, are uploaded into the server system (step 438). Entities from the user's documents, such as CAD entities from the CAD drawing and task entities from the

schedule, are identified (step 440). Activities are developed by associating task entities from uploaded schedule to CAD entities from the CAD drawing (step 442). In an exemplary embodiment, the developed activities form activity objects. A new schedule is generated based on the developed activities (step 444). At step 445, inter-relationships among the objects are developed and each object is stored in a main database in accordance with the developed inter-relationships.

GENERAL OPERATION

An exemplary system in accordance with the present invention comprises three functional layers: (1) a presentation layer; (2) an object layer; and (3) an interface and data access layer. The layers are inter-dependent and each layer includes different functionality. Briefly, the presentation layer organizes user interface components and defines the interaction between users and the system. The object layer categorizes information relating to physical and logical entities in a project. The interface and data access layer defines interfacing mechanisms to third-party software and database management systems. Functionality performed by each layer is discussed in more detail below.

The presentation layer provides interfaces between the system and the client. The client may be end users or Web sites. The presentation layer includes mechanism to gather information from clients as well as mechanism to present data to clients. In an exemplary embodiment, multiple interfaces are used to achieve the functionality. For example, input user interfaces and output interfaces gather data from clients as well as display and publish data to clients. Further, third-party user interfaces allow clients to input or review project data created by third-party software and object interfaces allow communication with objects in the object layer.

The object layer provides simple data validation logic as well as computing and data retrieving functionality. In an exemplary embodiment, each user interface can be mapped to one or more objects defined in the object layer. For instance, during the course of a project, a large number of physical and logical entities are typically developed. These entities can be organized into specific classes in the object layer. A class may comprise multiple attributes that delegate entity data and methods and define the behavior of the entities. An object, instantiated from its class, preserves

inherited attributes and methods that regulate the relationship between this object and other project objects. The relationship among objects maintains data integrity and consistency among objects.

An index of exemplary objects are as described in appendix A. Each exemplary object is identified by a numerical number and each attribute or method in an object is identified by an alphabetical letter. Numbers in brackets as shown in appendix A for each object represent the inter-relationship between the object to another object identified by that number. Similarly, a letter following a number in a bracket represents a corresponding attribute of the object identified by the number.

Exemplary objects in appendix A can be inter-related using a relational table structure, such as hash table structure. Figure 5 represents an exemplary inter-relationship of the objects listed in appendix A. As shown in Fig. 5, the inter-relationship among objects establishes a systematic mechanism for project data exchange among objects. Further, systematic data transaction maintains data integrity and achieves data compatibility among different software packages. In addition, the inter-relationship of the objects allow the object layer to coordinate data exchange between the presentation layer and the interface and data access layer.

The interface and data access layer defines interface mechanisms for data exchange between third-party software and objects in the object layer. In an exemplary embodiment, an interface between work area objects [object 3] and CAD software is provided. In this embodiment, CAD drawings may be uploaded from a client computer 106 or created within the system server 104. In any event, CAD entities in CAD drawings are linked to work area objects by assigning identifications ("IDs") to the CAD entities in the drawings. CAD entities represent the geometrical data of a work area object. When the geometrical data of a work area object is requested, corresponding CAD entities with matching IDs are accessed through the CAD's API. Similarly, when a work area object relating to a CAD entity is requested, all work area objects are checked to find the ones that match the CAD entity's ID.

In an exemplary embodiment, an interface between activity objects [object 15] and Scheduling/Project Management software is provided. In this embodiment, schedules created using Scheduling/Project Management software may be uploaded from a client computer 106 or created within the system server 104. In any event, task

entities in uploaded schedules are linked to activities objects by assigning identifications ("IDs") to the task entities in the schedules. Task entities represent scheduling data of an activity object. When scheduling data of an activity object is requested, corresponding task entities with matching IDs are accessed through the scheduling/project management software's API. Similarly, when an activity object relating to a task entity is requested, all activity objects are checked to find the ones that match the task entity's ID.

In yet another exemplary embodiment, interface between cost account objects [object 13] and cost estimating software is provided. In this embodiment, cost estimates created using cost estimating software may be uploaded from a client computer 106 or created within the system server 104. In any event, account entities in the uploaded schedules are linked to cost account objects by assigning identifications ("IDs") to the account entities in the cost estimates. Account entities represent detailed cost data of a cost account object. Thus, detailed cost data of a cost account object is bi-directionally linked to account entities by the IDs.

In another exemplary embodiment, an interface between simulation report object [object 21] and report creating software is provided. In this embodiment, reports created using report creating software may be uploaded from a client computer 106 or created within the system server 104. In any event, cell entities in uploaded reports are linked to simulation report objects by assigning identifications ("IDs") to the cell entities in the reports. Thus, cell entities of a simulation report object is bi-directionally linked to cell entities by the IDs. In simulation report objects, additional methods defined within the object, such as SET_X and SET_Y methods, may be invoked to further establish direct access to cell data in a third-party report creating software [object 21, attributes D, E].

In yet another exemplary embodiment, an interface to database management systems is provided. The persistence of objects in the object layer are maintained by the WRITE() and READ() method of each object (see appendix A). For example, when the WRITE() method of a current project object is invoked, the project object and all related/subordinate objects are invoked through recursive calling to corresponding WRITE() methods in those related/subordinate objects. Similarly, the

READ() method of the current project object can retrieve all related/subordinate objects from local file systems, database or remote nodes.

To further illustrate the inter-relationship between objects set forth in appendix A, in an exemplary method a user signs on to the system server 104, a new operator object [object 23] is created and added into the operator objects group [object 24]. The new operator object's LOGIN() method invokes a new session object [object 27]. Further, the user can associate a domain object [object 25] to the new operator object. The data access control table of the domain object [object 25, attribute C] regulates the user's right to access other objects.

Once a new operator object is created, the user is prompted to provide project information. Based on the provided information, a project class object [object 1] is initialized and added into the project objects group [object 2]. Next, the user is prompted to identify defined work areas. Based on the number of above ground and underground levels of the initiated project object, the system automatically generates a work area object [object 3]. The work area object defines floor plans for the new project. A user having appropriate access privileges can modify the generated floor plans or define new floor plans. The elevation attributes of the work area object [attribute E] and any successor work area objects [attribute C] determine the sequence of work area objects. In consideration of gravity, work area objects having the lowest elevation are usually created first. However, users having appropriate access privileges can reorganize the sequence of work area objects.

Next, according to the type of building structure defined in the current project object [object 1, attribute J], default building block objects [object 11] for the type of building structure are assigned or generated. Users having appropriate access privileges are allowed to review, add, delete or modify the default building block objects. Next, building element objects [object 17] and activity objects [object 15] are developed and linked.

In an exemplary embodiment, a user provides pre-existing CAD drawings and/or schedules. The user's pre-existing documents are uploaded into the system. CAD entities in any CAD drawings are identified. For each identified CAD entity, the user needs to identify related work areas, building elements, and building blocks. If such information is not discernible visually on the CAD drawings, a request for

additional information may be issued to the user for clarification. A new object for each identified building element is developed. The developed building element objects are associated with their related building block and work area objects.

Each building block object is related to a set of action objects and building element objects. An activity object can be generated based on the action objects and building element objects associated with a building block object. Each building element object includes the IDs of activity objects for that building element object. Similarly, each activity object includes the IDs of the related building element object for that activity object. As a result, a bi-directional relationship between an activity object and a building element object is established. Each action object [object 9] includes a list of resource usage objects [object 9, attribute E]. Thus, resource usage objects for each activity object can be gathered from associated action objects.

Activity objects are grouped based on the following: (1) the attribute in building element objects [object 17, attribute C] that specifies the IDs of associated work area objects; (2) the attribute in building element objects [object 17, attribute D] that specifies the IDs of associated building block objects; and (3) the attribute in activity objects [object 15, attribute F] that specifies associated action objects. For each category, a summary activity object group is developed. The summary activity object group includes activity objects having the same attribute category. A user can associate each instance of summary activity object group to one or more task entities in an existing schedule.

In another exemplary embodiment, a user provides pre-existing CAD drawings only. The system in accordance with an embodiment of the present invention generates schedules based on any existing drawings uploaded from the user. The user's pre-existing CAD drawings are uploaded into the system. The uploaded CAD drawings are processed in the same way as described above, whereby CAD entities in any CAD drawings are identified, building element objects are created, and created building element objects are associated with their related building block and work area objects. Further, in accordance with the process described above, activity objects are also developed and grouped according to attribute categories. When the activity objects are developed and grouped, a schedule can be generated. The summary activity object groups can be organized as follows: (1) the sequence of a work area object [object 3,

attribute C] associated with a building element object [object 17, attribute C] and child activities [object 15, attribute E]; (2) the elevation [object 3, attribute E] of building element objects [object 15, attribute C] of the work area object; (3) the sequence of CAD entities identified; and (4) the sequence of action objects associated with each identified CAD entity [object 15, attribute F]. Based on the above organizations, the system generates a new schedule using the summary activity object groups through the APIs of the scheduling/project management software. The generated schedule can be modified by users with appropriate access privileges.

In yet another exemplary embodiment, a user creates new drawings and schedules using the system in accordance with this invention. The user is prompted to identify a current work area and select a target building block object from a customized list provided through a user interface. Each item on the customized list provides a link to a pre-drawn CAD entity. Thus, when a target building block is selected, the linked CAD entity is drawn onto a CAD document. The user is allowed to select as many building blocks as desired to create a CAD drawing. After a CAD drawing is created, the system generates objects based on the CAD entities as described in the above embodiments. When the appropriate objects are developed, the system generates a schedule as described in the above embodiment.

Project data, either provided via CAD drawings or schedules, are encapsulated in target objects. Target objects can further be grouped into object groups. A user is able to access the encapsulated data for a project across a network, such as the Internet. Data encapsulated by each target object are referred to as instances. In general, changes to object instances that are encapsulated by target objects are distributed throughout the object groups. Thus, the present invention avoids data inconsistency and data re-entry.

In an exemplary embodiment, a user can obtain a display of activity information associated with a selected CAD entity. When a user selects one or more CAD entities from a CAD drawing, associated building element instances are highlighted by implementing a building object's HIGHLIGHT() method [object 17, attribute G]. Further, activity instances (including task entities) associated with the highlighted building element instances are highlighted by the HIGHLIGHT() method [object 15, attribute I]. Thus, direct links between CAD entities in the drawing and

task entities in the schedules are created and the user can review activity information by selecting CAD entities from the CAD drawing.

In another exemplary embodiment, a user can obtain resource information related to a selected CAD entity. When a user selects one or more CAD entities from a drawing, associated building element instances are highlighted by implementing building object's HIGHLIGHT() method [object 17, attribute G]. Activity instances associated with the highlighted building element instances are highlighted by the HIGHLIGHT() method. Further, resource entities associated to action instances that are linked to the highlighted activity instances are displayed. Thus, direct links between CAD entities in the drawing and resource entities are created and the user can review linked resource information by selecting CAD entities from the drawing.

In yet another exemplary embodiment, a user can highlight related building element instances in CAD drawings by selecting task entities. When one or more task entities from a schedule are selected, activity instances relating to the selected task entities are retrieved. The CAD entities representing the building element instances linked in the retrieved activity instances are highlighted by the HIGHLIGHT() method [object 17, attribute E].

In another exemplary embodiment, a user can obtain a display of related building element instances at any given time along a time line. Based on a time period specified by a user, the task entities for the specified time period are retrieved. Corresponding CAD entities of the related building element instances for the retrieved task entities (encapsulated in activity objects) are highlighted by the HIGHLIGHT() method. Thus, the user can observe the building element instances that are involved at any given time period.

In yet another exemplary embodiment, a user can obtain a display of building element instances in a critical path. When a user requests a display of building element instances in a critical path, task entities in the critical path are retrieved. CAD entities corresponding to the retrieved task entities are highlighted by the HIGHLIGHT() method. Thus, the user can distinguish building element instances in a critical path.

In another exemplary embodiment, a user can obtain a display of building element instances using selected resource. When an instance of resource entity

encapsulated in a resource object is selected, activity instances whose related action instances use the selected resource instance are retrieved. CAD entities of building element instances relating to the retrieved activity instances are highlighted. Thus, the user is able to view all CAD entities implementing the selected resource instance.

In yet another exemplary embodiment, a user can obtain a display of activities using selected resources. When an instance of a resource entity encapsulated in a resource object is selected, activity instances whose related action instances use the selected resource instance are retrieved. Task entities linked to the retrieved activity instances are highlighted. Thus, the user is able to view all task entities implementing the selected resource instance.

In another exemplary embodiment, a user can exchange data between drawings and resources. When a user selects one or more CAD entities from a drawing, building element instances corresponding to the selected CAD entities are highlighted. Further, activity instances related to the highlighted building element instances are also highlighted. Resource instances included in action instances of the highlighted activity instances are displayed. If dimensions of the selected CAD entities are modified, the number of linked resource usage instances can be updated by implementing the CALCULATE_VOLUME() method or the CALCULATE_LENGTH() method of the building element object [object 17, attribute K and L].

In yet another exemplary embodiment, a user can obtain a display of resource information in a work area. When an instance of work area is selected, related building element instances are retrieved. CAD entities related to the retrieved building element instances are highlighted. Resource usage information of the action instances relating to the activity instances that define the construction of the retrieved building element instances is displayed.

Construction projects can be implemented efficiently using systems provided in this invention. Advantages include efficient visualization of construction sequence, schedule tracing, cost projection, and resource allocation planing for a construction project.

In an exemplary embodiment, daily event instances [object 19] are used to provide project simulations. First, the system loops through daily event instances for each day of a project from its start date to its end date. For each project date, a daily

event object is developed. The date attribute of the developed daily event object [object 19, attribute B] is the looped date for its corresponding daily event instance. The daily event object also includes IDs of activity instances that should take place within the looped date [object 19, attribute C]. Further, instances of daily event objects are sorted in the order of their date attribute [object 19, attribute B] and maintained by the current daily events object group [object 20]. Based on a specified simulation date, daily event instances beginning from the project start date to the specified simulation date are used to prepare various simulations.

In one embodiment, construction sequence can be visualized in a simulation. The daily event instances during the simulation period include associations with appropriate activity instances. Further, the associated activity instances include association with appropriate building element instances. The DISPLAY() method of the involved building element instances are implemented to simulation the work that should be performed during the simulation period. In an exemplary embodiment, different colors are used in the simulation to represent different phases of the building element instances.

In another embodiment, project progress can be simulated for the specified simulation period. A progress line in a schedule for the project can be simulated for the specified simulation period by using the APIs of third-party scheduling/project management software.

In yet another embodiment, projected cash flow simulation is provided for the specified simulation period. An array of X-axis data in a simulation report object [object 21, attribute F] records the simulated day and an array of Y-axis data in the simulation report object [object 21, attribute G] records daily spending. Daily spending can be gathered from the balance attribute of a linked cost account instances [object 13, attribute C]. The latest daily or cumulative cash flow projections are published by implementing the SET_X() [object 21, attribute D] and SET_Y() [object 21, attribute E] methods.

In yet another exemplary embodiment, projected resource usage can be simulated for the specified simulation period. An array of X-axis data in a simulation report object [object 21, attribute F] records the simulated day and an array of Y-axis data in the simulation report object [object 21, attribute G] records the consumed

quantity of a resource instance. The latest resource usage projections are published by implementing the SET_X() [21, D] and SET_Y() [21, E] methods.

Changes to instances encapsulated by respective target objects can be incorporated into simulations by developing a new set of daily event instances based on the changes. As a result, a user can simulate project progress using various schedules and resource allocation.

The foregoing examples illustrate certain exemplary embodiments of the invention from which other embodiments, variations, and modifications will be apparent to those skilled in the art. The invention should therefore not be limited to the particular embodiments discussed above, but rather is defined by the following claims.

APPENDIX A

1. Project

Generic project information can be encapsulated into an instance of project class and its subordinate instances. An exemplary project instance may include, but is not limited to the following attributes and methods:

A. unique ID
B. name
C. description
D. start date
E. total cost
F. owner information
G. type of contracting method, which regulates the relationship between project participants and the data access control for each party.
H. number of ground levels
I. number of underground levels
J. type of building structure
K. an instance of building blocks [12], which manages a set of building block instances [11] used in this project.
L. an instance of building elements [18], which manages a set of building element instances [17] belonging to the current instance of this project class [1].
M. an instance of activities [16], which manages a set of activity instances [15] that denote tasks of the current project instance [1].
N. an instance of work areas [4], which manages a set of work area instances [3] belonging to the current project instance [1].
O. an instance of resources [6], which manages a set of resource instances [5] used by the current instance of this project class [1].
P. an instances of cost accounts [14], which manages a set of cost account instances [13] of the current project instance [1].
Q. an instances of daily events [20], which manages a set of daily event instances [19] belong to the current project instance [1].
R. an instance of simulation reports [22], which manages a set of simulation report [21] used in the current project instance [1].
S. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
T. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
U. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
V. CREATE() method, which instantiates an object of this class.
W. DESTROY() method, which removes the calling object from the invented systems.

2. Projects

The instance of projects class [2] groups a set of project instances [1] together. An exemplary projects class includes, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of project instances [1], which are managed by the current projects instance [2].
C. ADD(ID of project instance [1]) method, which adds the project instance [1], whose ID is passed as the parameter, into the array [2, B].
D. REMOVE(ID of project instance [1]) method, which removes the project instance [1], whose ID is passed as the parameter, from the array [2, B].
E. COUNT() method, which returns the number of instances in the ID array [2, B].
F. IS_MEMBER(ID of project instance [1]) method, which return a Boolean value representing whether the project instance [1], whose ID is passed as the parameter, is a member of the ID array [2, B] or not.
G. MEMBER(Index of the ID array [2, B]), which returns a project instance [1] , whose index number in the ID array [2, B] matches the passed one.
H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

3. Work Area

The instances of work area class [3] divide the physical space of the project site into manageable logical zones. An exemplary work area instance [3] may include but is not limited to the following attributes and methods.

A. unique ID
B. name
C. IDs of successor work area instances [3], most of whose building element instances [9] are created after these in this work area.
D. links to CAD entities, which represent the geometrical appearance of this work area in the linked CAD system [I].
E. elevation, which can be derived from the above CAD entities [3, D].
F. an instance of building elements [18], which manages building element instances [17] in this work area [3].
G. ID of project instance [1], which owns this work area instance [2].

H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

4. Work Areas

The instance of work areas class [4] groups a set of work area instances [3] together. An exemplary instance of work areas class includes, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of work area instances[3], which are managed by the current work areas instance [4].
C. ADD(ID of work area instance [3]) method, which adds the work area instance [3], whose ID is passed as the parameter, into the array [4, B].
D. REMOVE(ID of work area instance [3]) method, which removes the work area instance [3], whose ID is passed as the parameter, from the array [4, B].
E. COUNT() method, which returns the number of instances in the ID array [4, B].
F. IS_MEMBER(ID of work area instance [3]) method, which returns a Boolean value representing whether the work area instance [3], whose ID is passed as the parameter, is a member of the ID array [4, B] or not.
G. MEMBER(Index of the ID array [4, B]), which returns a work area instance [3] , whose index number in the ID array [4, B] matches the passed one.
H. LINK_TO_MEMBER(Links to CAD entities), which returns a work area instance[3], whose links to the geometrical representation [3, D] of CAD system [I] matches the passed one.
I. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
J. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
K. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
L. CREATE() method, which instantiates an object of this class.
M. DESTROY() method, which removes the calling object from the invented systems.

5. Resource

Each resource used in the current project, including labor, equipment, material, contractors, could be organized by the resource class [5]. An exemplary resource instance [5] may include, but is not limited to, following attributes and methods.

A. unique ID
B. name
C. unit of measure, which is used to measure the quantity of the current class
D. type of resource, which can be Labor, Equipment, Material or Contractor.
E. cost per use, which is the sum of the setup fee and other cost for using this class
F. cost per unit, which is the cost for using a unit of the current class
G. effective date
H. productivity
I. ID of project instance [1], which owns this resource instance [5].
J. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
K. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
L. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
M. CREATE() method, which instantiates an object of this class.
N. DESTROY() method, which removes the calling object from the invented systems.

6. Resources

The instance of resources class [6] groups a set of resource instances [5] together. An exemplary resources class includes, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of resource instances[5], which are managed by the current resources instance [6].
C. ADD(ID of resource instance [5]) method, which adds the resource instance [5], whose ID is passed as the parameter, into the array [6, B].
D. REMOVE(ID of resource instance [5]) method, which removes the resource instance [5], whose ID is passed as the parameter, from the array [6, B].
E. COUNT() method, which returns the number of instances in the ID array [6, B].
F. IS_MEMBER(ID of resource instance [5]) method, which returns a Boolean value representing whether the instance, whose ID is passed as the parameter, is a member of the ID array [6, B] or not.
G. MEMBER(Index of the ID array [6, B]), which returns a resource instance [5] , whose index number in the ID array [6, B] matches the passed one.
H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.

I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

7. Resource Usage

This class organizes the information of resource use. An exemplary resource usage instance [7] may include, but is not limited to, the following attributes and methods.

A. unique ID
B. ID of involved resource instance [6]
C. quantity of consumed resource
D. ID of involved action instance [9], which requires this resource usage instance [7].
E. ID of cost account instance [13], which the cost of this resource usage instance [7] reports to.
F. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
G. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
H. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
I. CREATE() method, which instantiates an object of this class.
J. DESTROY() method, which removes the calling object from the invented systems.

8. Resource Usages

The instance of resource usages class [8] groups a set of resource usage instances [7] together. An exemplary resource usages class includes, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of resource usage instances[7], which are managed by the current resource usages instance [8].
C. ADD(ID of resource usage instance [7]) method, which adds the resource usage instance [7], whose ID is passed as the parameter, into the array [8, B].
D. REMOVE(ID of resource usage instance [7]) method, which removes the resource usage instance [7], whose ID is passed as the parameter, from the array [8, B].
E. COUNT() method, which returns the number of instances in the ID array [8, B].

F. IS_MEMBER(ID of resource usage instance [7]) method, which returns a Boolean value representing whether the resource usage instance [7], whose ID is passed as the parameter, is a member of the ID array [8, B] or not.
G. MEMBER(Index of the ID array [8, B]), which returns a resource usage instance [7], whose index number in the ID array [8, B] matches the passed one.
H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

9. Action

Each tasks for a building block instance [6] can be decomposed into a set of action instances, which may include, but is not limited to, the following exemplary attributes and methods.

A. unique ID
B. name
C. duration
D. denoted color to paint the linked CAD entities [17, E] of involved building element instances [17], which perform this action.
E. Instance of resource usages [8], which manages resource usage instance [7] defining the type and amount of resource used by the current action instance [9].
F. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
G. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
H. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
I. CREATE() method, which instantiates an object of this class.
J. DESTROY() method, which removes the calling object from the invented systems.

10. Actions

The instance of actions class [10] groups a set of action instances [9] together. An exemplary actions class includes, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of action instances[9], which are managed by the current actions instance [10].
C. ADD(ID of action instance [9]) method, which adds the action instance [9], whose ID is passed as the parameter, into the array [10, B].
D. REMOVE(ID of action instance [9]) method, which removes the action instance [9], whose ID is passed as the parameter, from the array [10, B].
E. COUNT() method, which returns the number of instances in the ID array [10, B].
F. IS_MEMBER(ID of action instance [9]) method, which returns a Boolean value representing whether the action instance [9], whose ID is passed as the parameter, is a member of the ID array [10, B] or not.
G. MEMBER(Index of the ID array [10, B]), which returns a action instance [9] , whose index number in the ID array [10, B] matches the passed one.
H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

11. Building Block

This class is the template for development of building element instances [17]. An exemplary building block class may include, but is not limited to, the following attributes and methods.

A. unique ID
B. name
C. links to denoted CAD entities, which represent the physical appearance of this building block in the CAD drawing.
D. IDs of child building block instances [11]. A building block instance may consist of none or multiple other building blocks.
E. an instance of actions [10], which manages action instance [9] defining the tasks for the completion of the current building block instance [11].
F. ID of project instance [1], which include this building block instance [11].
G. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
H. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.

I. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
J. CREATE() method, which instantiates an object of this class.
K. DESTROY() method, which removes the calling object from the invented systems.

12. Building Blocks

The instance of building blocks class [10] groups a set of building block instances [9] together. An exemplary building blocks class includes, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of building block instances[11], which are managed by the current building blocks instance [12].
C. ADD(ID of building block instance [11]) method, which adds the building block instance [11], whose ID is passed as the parameter, into the array [12, B].
D. REMOVE(ID of building block instance [11]) method, which removes the building block instance [11], whose ID is passed as the parameter, from the array [12, B].
E. COUNT() method, which returns the number of ID in the array [12, B].
F. IS_MEMBER(ID of building block instance [11]) method, which returns a Boolean value representing whether the building block instance [11], whose ID is passed as the parameter, is a member of the ID array [12, B] or not.
G. MEMBER(Index of the ID array [12, B]), which returns a building block instance [11] , whose index number in the ID array [12, B] matches the passed one.
H. LINK_TO_MEMBER(Link to 3-party CAD entities), which returns a building block instance[11], whose link to the geometrical representation [3, D] of CAD system [I] matches the passed one.
I. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
J. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
K. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
L. CREATE() method, which instantiates an object of this class.
M. DESTROY() method, which removes the calling object from the invented systems.

13. Cost Account

The cost and budget information are organized by a set of cost account instances, which may include, but is not limited to, the following attributes and methods.

A. unique ID
B. name

C. balance, which is the amount of capital in this account
D. currency
E. links to corresponding account entities in third-party cost estimating or accounting software [III].
F. ID of project instance [1], which include the current cost account instance [13].
G. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
H. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
I. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
J. CREATE() method, which instantiates an object of this class.
K. DESTROY() method, which removes the calling object from the invented systems.

14. Cost Accounts

The instance of cost accounts class [14] groups a set of cost account instances [13] together. An exemplary cost accounts class includes, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of cost account instances[13], which are managed by the current cost accounts instance [14].
C. ADD(ID of cost account instance [13]) method, which adds the cost account instance [13], whose ID is passed as the parameter, into the array [14, B].
D. REMOVE(ID of action instance [13]) method, which removes the cost account instance [13], whose ID is passed as the parameter, from the array [14, B].
E. COUNT() method, which returns the number of instances in the ID array [14, B].
F. IS_MEMBER(ID of cost account instance [13]) method, which returns a Boolean value representing whether the cost account instance [13], whose ID is passed as the parameter, is a member of the ID array [14, B] or not.
G. MEMBER(Index of the ID array [14, B]), which returns a cost account instance [13] , whose index number in the ID array [14, B] matches the passed one.
H. LINK_TO_MEMBER(Link to 3-party account entities), which returns a cost account instance[13], whose link to the geometrical representation [3, D] of CAD system [I] matches the passed one.
I. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
J. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
K. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.

L. CREATE() method, which instantiates an object of this class.
M. DESTROY() method, which removes the calling object from the invented systems.

15. Activity

This class manages the tasks for the completion of the building element instances [17]. An exemplary activity class may include, but is not limited to, the following attributes and methods.

A. unique ID
B. name
C. ID of related building element instances [17]
D. a Boolean value to determine whether this class is a summary activity or not.
E. IDs of child activity instances [15]
F. ID of action instance [9], which are included by this activity instance [15].
G. links to corresponding task entities in the schedule of the third-party scheduling and project management system [II].
H. CPM/PERT attributes, which may be derived from the linked task entities [15, G].
I. HIGHLIGHT() method, which change the color of linked task entity [15, G] to highlighted color through the API of the scheduling or project management system [II].
J. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
K. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
L. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
M. CREATE() method, which instantiates an object of this class.
N. DESTROY() method, which removes the calling object from the invented systems.

16. Activities

The instance of activities class [16] groups a set of activity instances [15] together. An exemplary activities class includes, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of activity instances[15], which are managed by the current activities instance [16].
C. ADD(ID of activity instance [15]) method, which adds the activity instance [15], whose ID is passed as the parameter, into the array [16, B].
D. REMOVE(ID of activity instance [15]) method, which removes the activity instance [15], whose ID is passed as the parameter, from the array [16, B].

E. COUNT() method, which returns the number of instances in the ID array [16, B].
F. IS_MEMBER(ID of activity instance [15]) method, which returns a Boolean value representing whether the activity instance [15], whose ID is passed as the parameter, is a member of the ID array [16, B] or not.
G. MEMBER(Index of the ID array [16, B]), which returns a activity instance [15] , whose index number in the ID array [16, B] matches the passed one.
H. LINK_TO_MEMBER(Link to 3-party task entities), which returns an activity instance[15], whose link to the corresponding task entities [15, G] of scheduling/project management system [II] matches the passed one.
I. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
J. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
K. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
L. CREATE() method, which instantiates an object of this class.
M. DESTROY() method, which removes the calling object from the invented systems.

17. Building Element

This class represents a physical project object, which occupies work area [3], performs activity [15] and consumes resource [5]. An exemplary building element instance may include, but is not limited to the following attributes and methods.

A. unique ID
B. name
C. ID of work area instance [3], where this building element resides
D. ID of associated building block instance [11], which is used as a template for the creation of the current instance.
E. links to the CAD entities, which defines the geometrical appearance of the current instance in the CAD system [I].
F. IDs of activity instances [15], which represents the tasks for the completion of this building element.
G. HIGHLIGHT() method, which changes the color of the linked CAD entity [17, E] to the highlighted color through CAD API.
H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.

K. CACULATE_VOLUME() method, which returns the current volume of the linked CAD entities [17, E].
L. CACULATE_LENGTH() method, which returns the current length of the linked CAD entities [17, E].
M. DISPLAY(ID of Action) method, which changes the color of the linked CAD entities [17, E] to the denoted color [9, D] of the passed action instance [9].
N. CREATE() method, which instantiates an object of this class.
O. DESTROY() method, which removes the calling object from the invented systems.

18. Building Elements

The instance of building elements class [18] groups building element instances [17] together. An exemplary building element class may include, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of building element instances[17], which are managed by the current building elements instance [18].
C. ADD(ID of building element instance [17]) method, which adds the building element instance [17], whose ID is passed as the parameter, into the array [18, B].
D. REMOVE(ID of action instance [17]) method, which removes the building element instance [17], whose ID is passed as the parameter, from the array [18, B].
E. COUNT() method, which returns the number of instances in the ID array [18, B].
F. IS_MEMBER(ID of building element instance [17]) method, which returns a Boolean value representing whether the building element instance [17], whose ID is passed as the parameter, is a member of the ID array [18, B] or not.
G. MEMBER(Index of the ID array [18, B]), which returns a building element instance [17] , whose index number in the ID array [18, B] matches the passed one.
H. LINK_TO_MEMBER(Link to 3-party CAD entities), which returns a building element instance[17], whose link to the geometrical representation [3, D] of CAD system [I] matches the passed one.
I. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
J. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
K. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
L. CREATE() method, which instantiates an object of this class.
M. DESTROY() method, which removes the calling object from the invented systems.

In addition to the above classes, the instances of following exemplary classes organize project data chronically, which can be used to simulate the construction sequence. It is understood

that the following list of classes is by no means inclusive, but is only intended to provide examples that may be incorporated for use into the present invention:

19. Daily Event

Project information for a given project day can be organized into a daily event. An exemplary daily event class may include, but is not limited to, the following attributes and methods:

A. unique ID
B. date, which represents when this event activates
C. ID of activity instances [15], whose period include the date [19, B] of this daily event instance [19].
D. IDs of simulation report instances [21], which employ the current daily event instance [19].
E. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
F. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
G. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
H. CREATE() method, which instantiates an object of this class.
I. DESTROY() method, which removes the calling object from the invented systems.

20. Daily Events

The instance of daily events class [20] groups a set of daily event instances [19] together. An exemplary daily event class may include, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of daily event instances[19], which are grouped by the current daily events instance [20].
C. ADD(ID of daily event instance [19]) method, which adds the daily event instance [19], whose ID is passed as the parameter, into the array [20, B].
D. REMOVE(ID of daily event instance [19]) method, which removes the daily event instance [19], whose ID is passed as the parameter, from the array [20, B].
E. COUNT() method, which returns the number of instances in the ID array [20, B].
F. IS_MEMBER(ID of daily event instance [19]) method, which returns a Boolean value representing whether the daily event instance [19], whose ID is passed as the parameter, is a member of the ID array [20, B] or not.
G. MEMBER(Index of the ID array [20, B]), which returns a daily event instance [19] , whose index number in the ID array [20, B] matches the passed one.

H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

21. Simulation Report

This class publishes the project information in the order of construction sequence. An exemplary simulation report instance may include, but is not limited to, the following attributes and methods. The instances of this class may be used to report cash flow projection, resource allocation and other chronical project control information.

A. unique ID
B. ID of related daily event instances [20]
C. links to the cell entities of a reporting system.
D. SET_X() method, which update X-axis data through the API of the linked reporting system [IV].
E. SET_Y() method, which update Y-axis data through the API of the linked reporting system [IV].
F. an array of X-axis data
G. an array of Y-axis data
H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

22. Simulation Reports

The instance of simulation reports class [21] groups a set of simulation report instances [19] together. An exemplary simulation report class may include, but is not limited to, the following attributes and methods.

A. unique ID

B. array of ID of simulation report instances[21], which are managed by the current simulation reports instance [22].
C. ADD(ID of simulation report instance [21]) method, which adds the simulation report instance [22], whose ID is passed as the parameter, into the array [22, B].
D. REMOVE(ID of simulation instance [21]) method, which removes the simulation report instance [21], whose ID is passed as the parameter, from the array [22, B].
E. COUNT() method, which returns the number of instances in the ID array [22, B].
F. IS_MEMBER(ID of simulation report instance [21]) method, which returns a Boolean value representing whether the simulation report instance [21], whose ID are passed as the parameter, is a member of the ID array [22, B] or not.
G. MEMBER(Index of the ID array [22, B]), which returns a simulation report instance [22] , whose index number in the ID array [22, B] matches the passed one.
H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

Lastly, administrative data of the implemented systems may be encapsulated into specific classes such as the ones documented below. It is understood that the following list of classes is by no means inclusive, but is only intended to provide examples that may be incorporated for use into the present invention:

23. Operator

The operator class denotes a user who initiates a session instance [27] and has interaction with instances of any classes defined in the invented model. An exemplary operator class may include, but is not limited to, the following attributes and methods.

A. unique ID
B. user name
C. password
D. effective date
E. IDs of domain instances [25], which the current instance belongs to
F. LOGIN() method, which create a new instance of session [27].
G. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
H. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.

I. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
J. CREATE() method, which instantiates an object of this class.
K. DESTROY() method, which removes the calling object from the invented systems.

24. Operators

The instance of operators class [24] groups a set of operator instances [23] together. An exemplary operators class may include, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of operator instances[23], which are managed by the current operators instance [24].
C. ADD(ID of operator instance [23]) method, which adds the operator instance [23], whose ID is passed as the parameter, into the array [24, B].
D. REMOVE(ID of operator instance [23]) method, which removes the operator instance [23], whose ID is passed as the parameter, from the array [24, B].
E. COUNT() method, which returns the number of instances in the ID array [24, B].
F. IS_MEMBER(ID of operator instance [23]) method, which returns a Boolean value representing whether the operator instance [23], whose ID is passed as the parameter, is a member of the ID array [24, B] or not.
G. MEMBER(Index of the ID array [24, B]), which returns an operator instance [23] , whose index number in the ID array [24, B] matches the passed one.
H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

25. Domain

Each operator instance [23] may be associated with different instances of the domain class, which may include, but is not limited to the following attributes and methods.

A. unique ID
B. name
C. the data access control table, which is used to determine whether the current users have privilege to create, retrieve, update or delete attributes of model instances.

D. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
E. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
F. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
G. CREATE() method, which instantiates an object of this class.
H. DESTROY() method, which removes the calling object from the invented systems.

26. Domains

The instance of domains class [26] groups a set of domain instances [25] together. An exemplary domains class may include, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of domain instances[25], which are managed by the current domains instance [26].
C. ADD(ID of domain instance [25]) method, which adds the domain instance [25], whose ID is passed as the parameter, into the array [26, B].
D. REMOVE(ID of domain instance [25]) method, which removes the domain instance [25], whose ID is passed as the parameter, from the array [26, B].
E. COUNT() method, which returns the number of instances in the ID array [26, B].
F. IS_MEMBER(ID of domain instance [25]) method, which returns a Boolean value representing whether the domain instance [25], whose ID is passed as the parameter, is a member of the ID array [26, B] or not.
G. MEMBER(Index of the ID array [26, B]), which returns a domain instance [25] , whose index number in the ID array [26, B] matches the passed one.
H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

13. Session

Software session created by operator instances [23] may be represented by this class, which may include, but is not limited to the following attributes and methods.

A. unique ID
B. name
C. machine name, on which the instance of the current session instance is invoked and executed
D. an instance of current operator [23] of this session instance [27].
E. ID of project instance [1], which is opened by the current session instance [27].
F. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
G. OPEN_PROJECT() method, which initializes a new project instance [1] or retrieves an existing project instance [1].

8. Sessions

The instance of sessions class [26] groups a set of session instances [25] together. An exemplary sessions class may include, but is not limited to, the following attributes and methods.

A. unique ID
B. array of ID of session instances[25], which are managed by the current sessions instance [26].
C. ADD(ID of session instance [25]) method, which adds the session instance [25], whose ID is passed as the parameter, into the array [26, B].
D. REMOVE(ID of session instance [25]) method, which removes the session instance [25], whose ID is passed as the parameter, from the array [26, B].
E. COUNT() method, which returns the number of instances in the ID array [26, B].
F. IS_MEMBER(ID of session instance [25]) method, which returns a Boolean value representing whether the session instance [25], whose ID is passed as the parameter, is a member of the ID array [26, B] or not.
G. MEMBER(Index of the ID array [26, B]), which returns a session instance [25] , whose index number in the ID array [26, B] matches the passed one.
H. audit information, including who creates the current instance of this class, when this instance was created, who updates this instance and when this instance was modified.
I. READ() method, which reads information of this instance from the local file system, database or network nodes. Whenever retrieving instances of subordinate classes, it triggers their READ() methods recursively.
J. WRITE() method, which saves information of this instance to the local file system, database or network nodes. Whenever saving instances of subordinate classes, it invokes their WRITE() methods recursively.
K. CREATE() method, which instantiates an object of this class.
L. DESTROY() method, which removes the calling object from the invented systems.

WHAT IS CLAIMED IS:

1. A method for processing data received across a network in multiple formats, comprising the steps of:

(a) receiving data at a server computer from a client computer;

(b) identifying a selected parser program based at least in part on said received data;

(c) parsing said received data using said selected parser program to create parsed data;

(d) selectively establishing assigned portions of said parsed data to a target object in a database;

(e) encapsulating said assigned portions within said target object to form a new target object; and

(f) storing said new target object in said database.

2. The method of claim 1, further comprising the steps of:

(g) receiving a request from said client computer;

(h) selectively retrieving data stored in said database in response to said request to form retrieved data;

(i) converting said retrieved data into a network compatible format to form converted data;

(j) generating a report based on said converted data; and

(k) sending said report to said client computer.

3. The method of claim 2, wherein said converting step includes the step of forming converted data in a HTML or XML format.

4. The method of claim 1, wherein said identifying step includes the step of identifying a selected parser program from a set of parser programs.

5. The method of claim 1, further comprising the step of inter-relating a set of target objects in a relational table structure.

6. A method for processing data received across a network in multiple formats, comprising the steps of:

(a) uploading a document provided by a client computer;

(b) identifying a selected parser program based at least in part on said document;

(c) parsing said document using said selected parser program to provide a parsed document;

(d) identifying an entity from said parsed document; and

(e) associating said entity to an object in a schedule.

7. The method of claim 6, wherein said uploading step includes uploading a CAD document.

8. A method for processing data received across a network in multiple formats, comprising the steps of:

(a) uploading a document p rovided by said client computer;

(b) identifying a selected parser program based at least in part on said document;

(c) parsing said document using said selected parser program to produce a parsed document;

(d) specifying an identified entity from said parsed document;

(e) developing an object based at least in part on said identified entity to form a developed object;

(f) grouping said developed object in an object group;

(g) generating a schedule based at least in part on said object group.

9. The method of claim 8, wherein said grouping in step (f) includes assigning said developed object to a defined class.

10. The method of claim 8, wherein said grouping in step (f) includes creating an inter-relationship between said developed object and an existing object based at least in part on attributes of said developed object.

11. The method of claim 10, wherein said creating step includes the step of creating said inter-relationship using a relational table structure.

12. A method for processing data received across a network in multiple formats, comprising the steps of:

(a) enabling a client computer to specify a defined work area;

(b) assigning a building block to said defined work area;

(c) producing a developed object based at least in part on said building block;

(d) grouping said developed object in an object group;

(e) generating a schedule based on said object group; and

(f) storing said at developed object, said object group, and said schedule in a database.

13. The method of claim 12, wherein said grouping in step (d) includes assigning said developed object to a defined class.

14. The method of claim 12, wherein said grouping in step (d) includes creating an inter-relationship between said developed object and an existing object based at least in part on attributes of said developed object.

15. The method of claim 14, wherein said creating step includes the step of using a relational table structure.

16. The method of claim 14, wherein said producing step includes producing a developed object with a building element object and an action object.

17. The method of claim 16, wherein said grouping in step (d) creates at least one activity object based on said building element object and said action object.

18. The method of claim 12, further comprising the steps of:

(g) receiving a request from said client computer;

(h) selectively retrieving data stored in said database in response to said request;

(i) converting said data into a network compatible format to form converted data;

(j) creating a generated report based on said converted data; and

(k) sending said generated report to said client computer.

19. A system for processing data received across a network in multiple formats, comprising:

a central processing unit;

a communication interface for connection between said network and said central processing unit; and

a memory;

said memory including a set of data parser programs and a database having a set of objects;

wherein data received through said communication interface is processed by said central processing unit utilizing at least one of said set of data parser programs to create processed data, said processed data being selectively associated with one of said set of objects in said database and stored in said database based on said association.

20. The system of claim 19, wherein said memory further comprises a report generator module; wherein said report generator module includes logic code to selectively produce retrieved data from said database in response to a request from a client computer across said network, logic code to convert said retrieved data into converted data in a network compatible format, and logic code to send said converted data across said network to said client computer.

21. The system of claim 19, wherein said network is a distributed network.

22. The system of claim 19, wherein said client computer includes an end user or a web site.

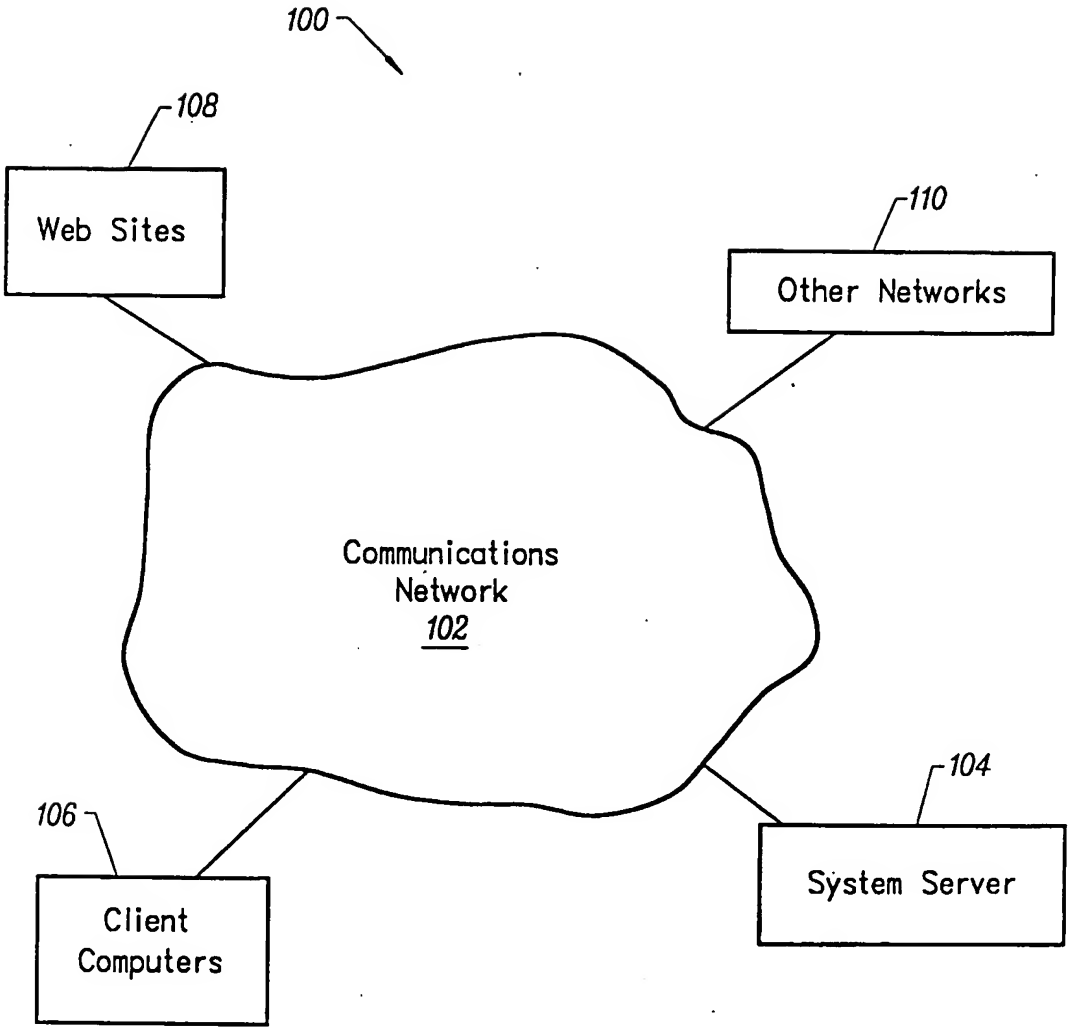

100
108
Web Sites
110
Other Networks
Communications
Network
102
104
System Server
106
Client
Computers

FIG. 1

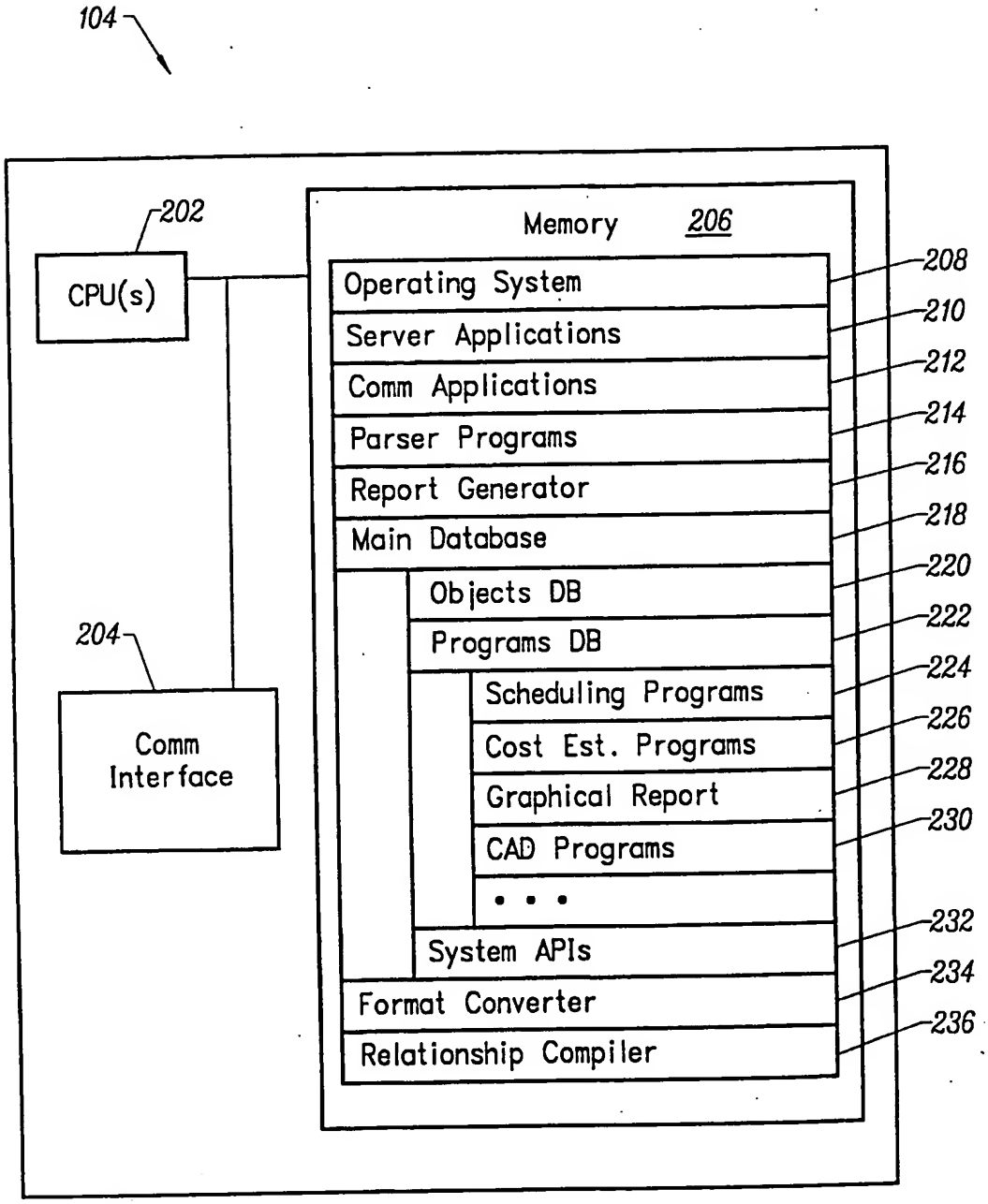

104
202
CPU(s)
204
Comm Interface
Memory 206
Operating System
Server Applications
Comm Applications
Parser Programs
Report Generator
Main Database
Objects DB
Programs DB
Scheduling Programs
Cost Est. Programs
Graphical Report
CAD Programs
• • •
System APIs
Format Converter
Relationship Compiler
208
210
212
214
216
218
220
222
224
226
228
230
232
234
236

FIG. 2A

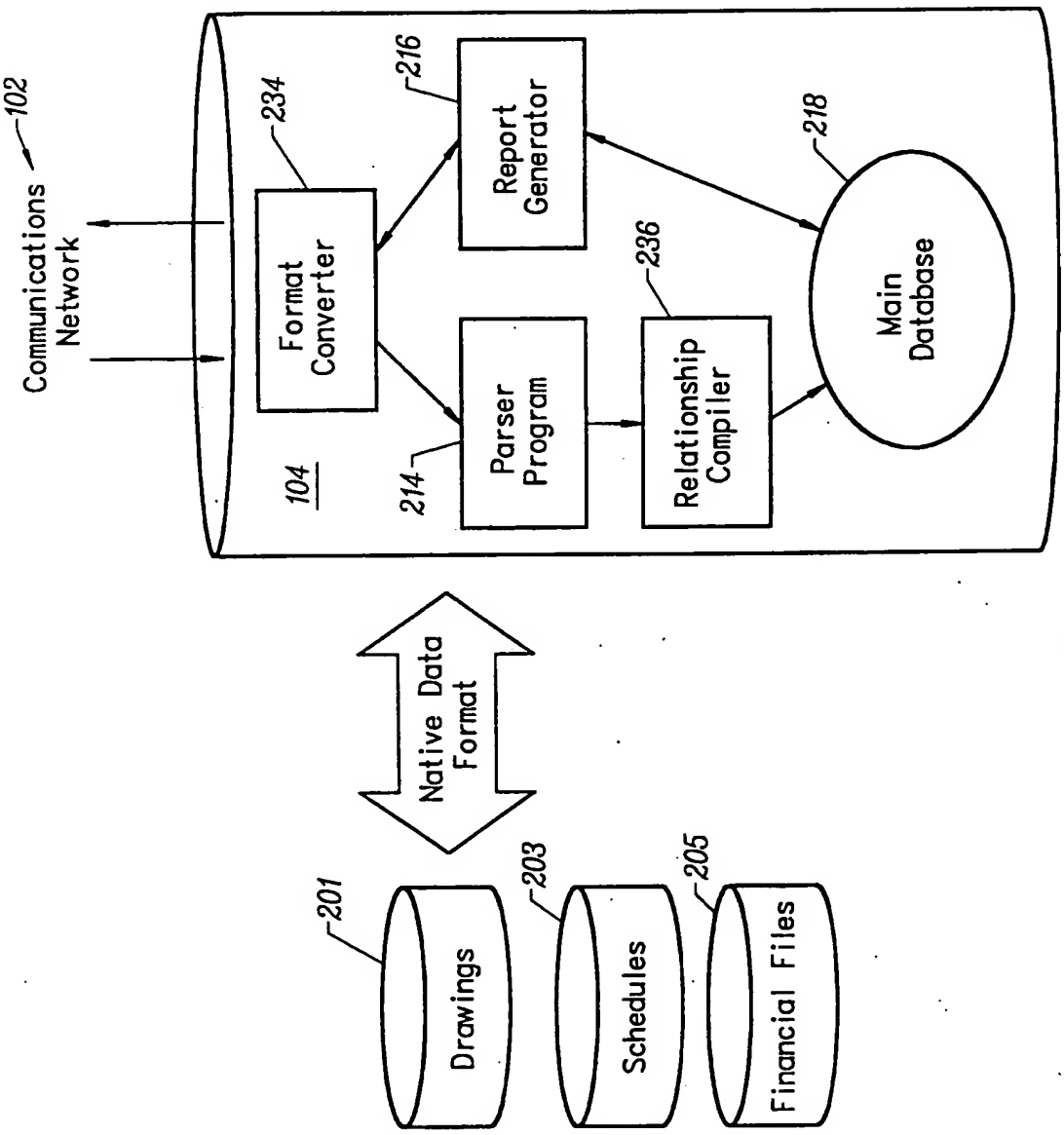

102
Communications Network
234
Format Converter
216
Report Generator
218
Main Database
236
Relationship Compiler
Parser Program
214
104
Native Data Format
201
Drawings
203
Schedules
205
Financial Files

FIG. 2B

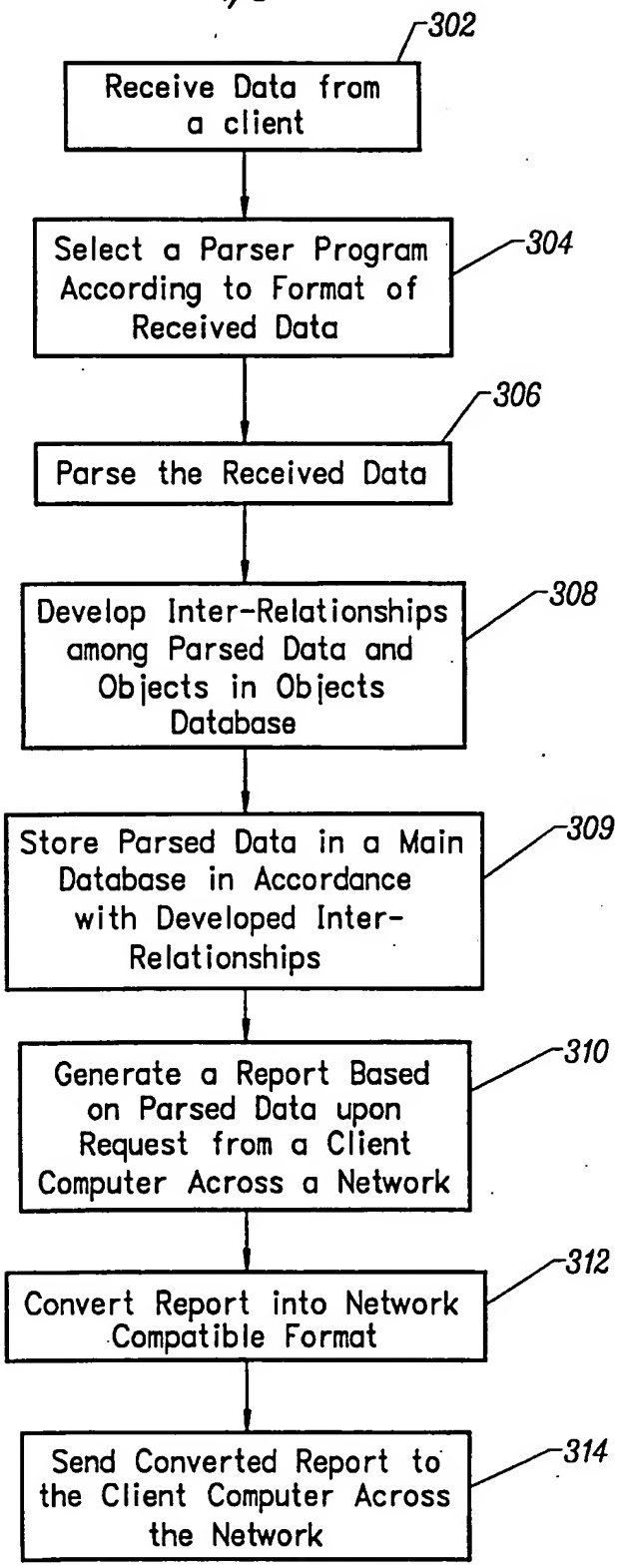

302
Receive Data from a client
Select a Parser Program According to Format of Received Data
304
306
Parse the Received Data
Develop Inter-Relationships among Parsed Data and Objects in Objects Database
308
Store Parsed Data in a Main Database in Accordance with Developed Inter-Relationships
309
Generate a Report Based on Parsed Data upon Request from a Client Computer Across a Network
310
Convert Report into Network Compatible Format
312
Send Converted Report to the Client Computer Across the Network
314

FIG. 3

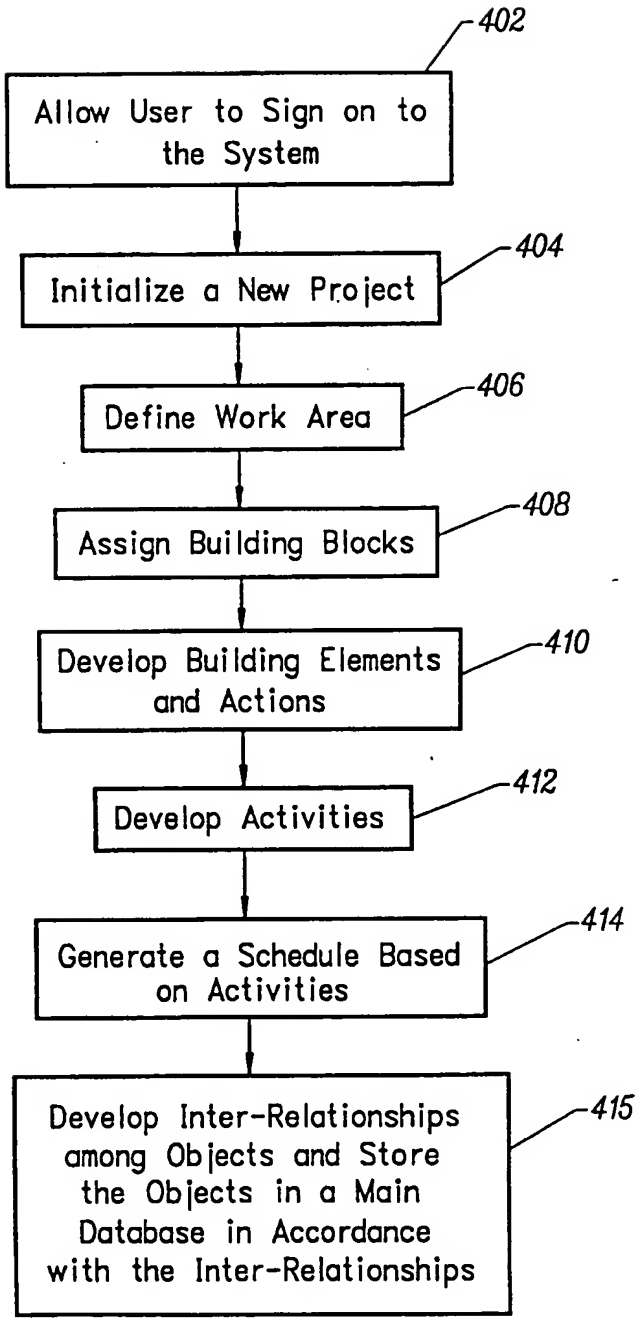

402
Allow User to Sign on to the System
404
Initialize a New Project
406
Define Work Area
408
Assign Building Blocks
410
Develop Building Elements and Actions
412
Develop Activities
414
Generate a Schedule Based on Activities
415
Develop Inter-Relationships among Objects and Store the Objects in a Main Database in Accordance with the Inter-Relationships

FIG. 4A

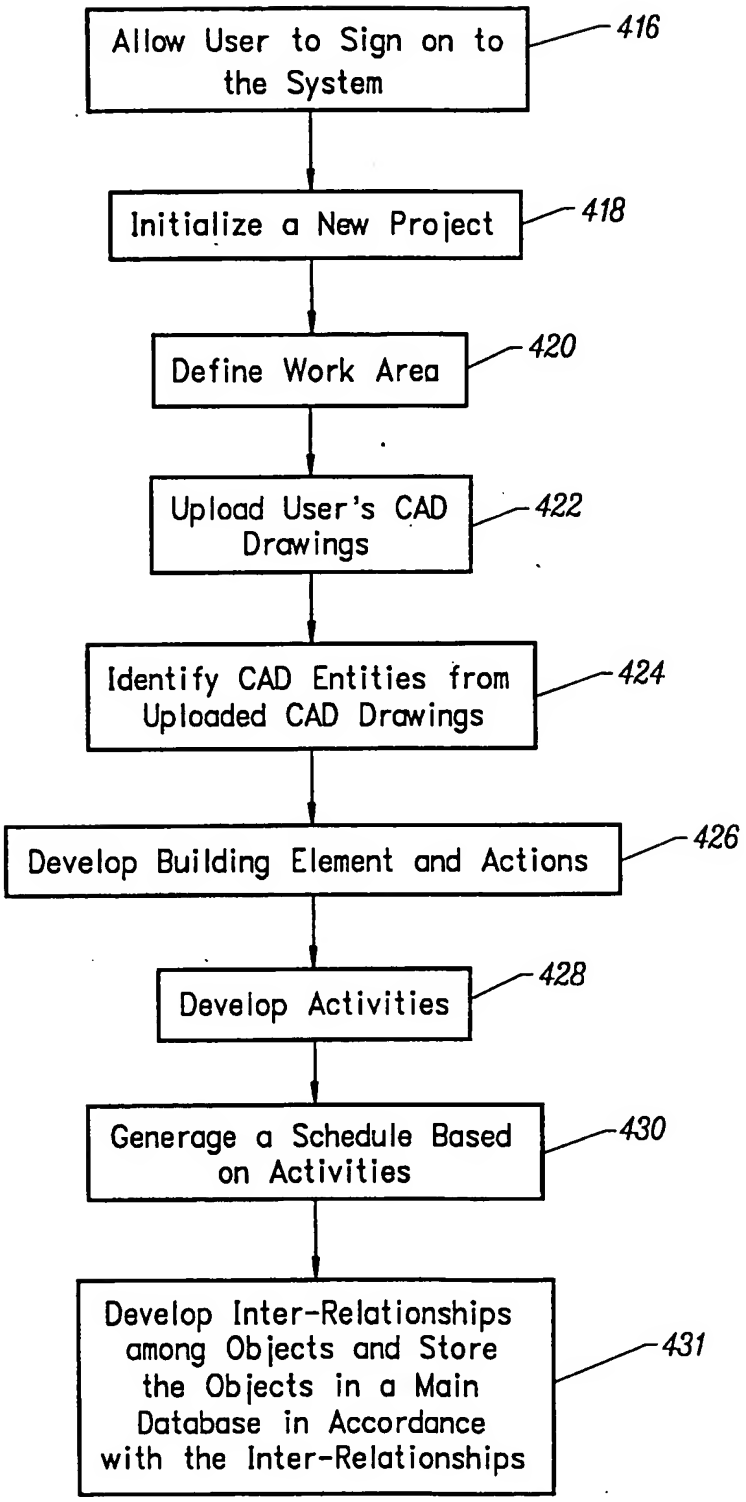

Allow User to Sign on to the System
416
Initialize a New Project
418
Define Work Area
420
Upload User's CAD Drawings
422
Identify CAD Entities from Uploaded CAD Drawings
424
Develop Building Element and Actions
426
Develop Activities
428
Generage a Schedule Based on Activities
430
Develop Inter-Relationships among Objects and Store the Objects in a Main Database in Accordance with the Inter-Relationships
431

FIG. 4B

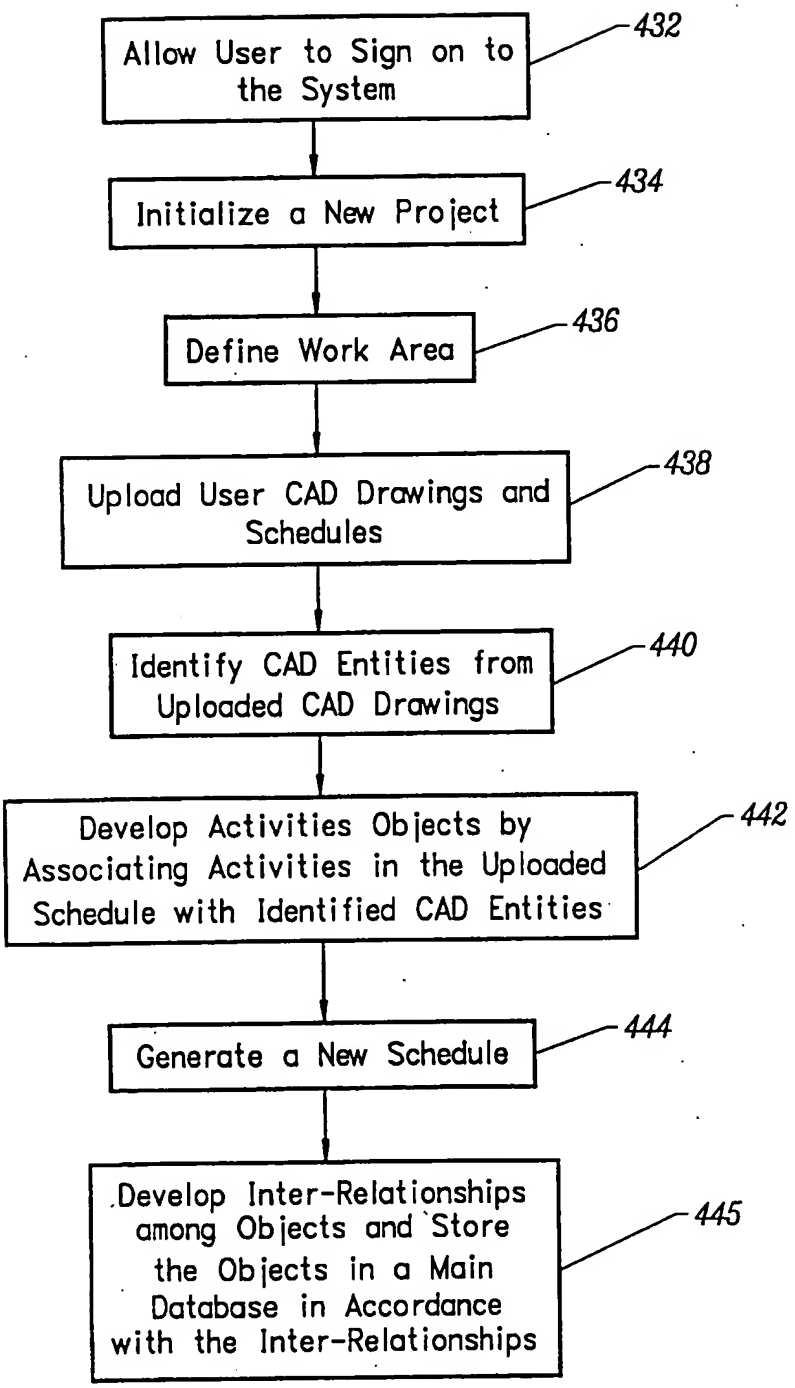

Allow User to Sign on to the System
432
Initialize a New Project
434
Define Work Area
436
Upload User CAD Drawings and Schedules
438
Identify CAD Entities from Uploaded CAD Drawings
440
Develop Activities Objects by Associating Activities in the Uploaded Schedule with Identified CAD Entities
442
Generate a New Schedule
444
Develop Inter-Relationships among Objects and Store the Objects in a Main Database in Accordance with the Inter-Relationships
445

FIG. 4C

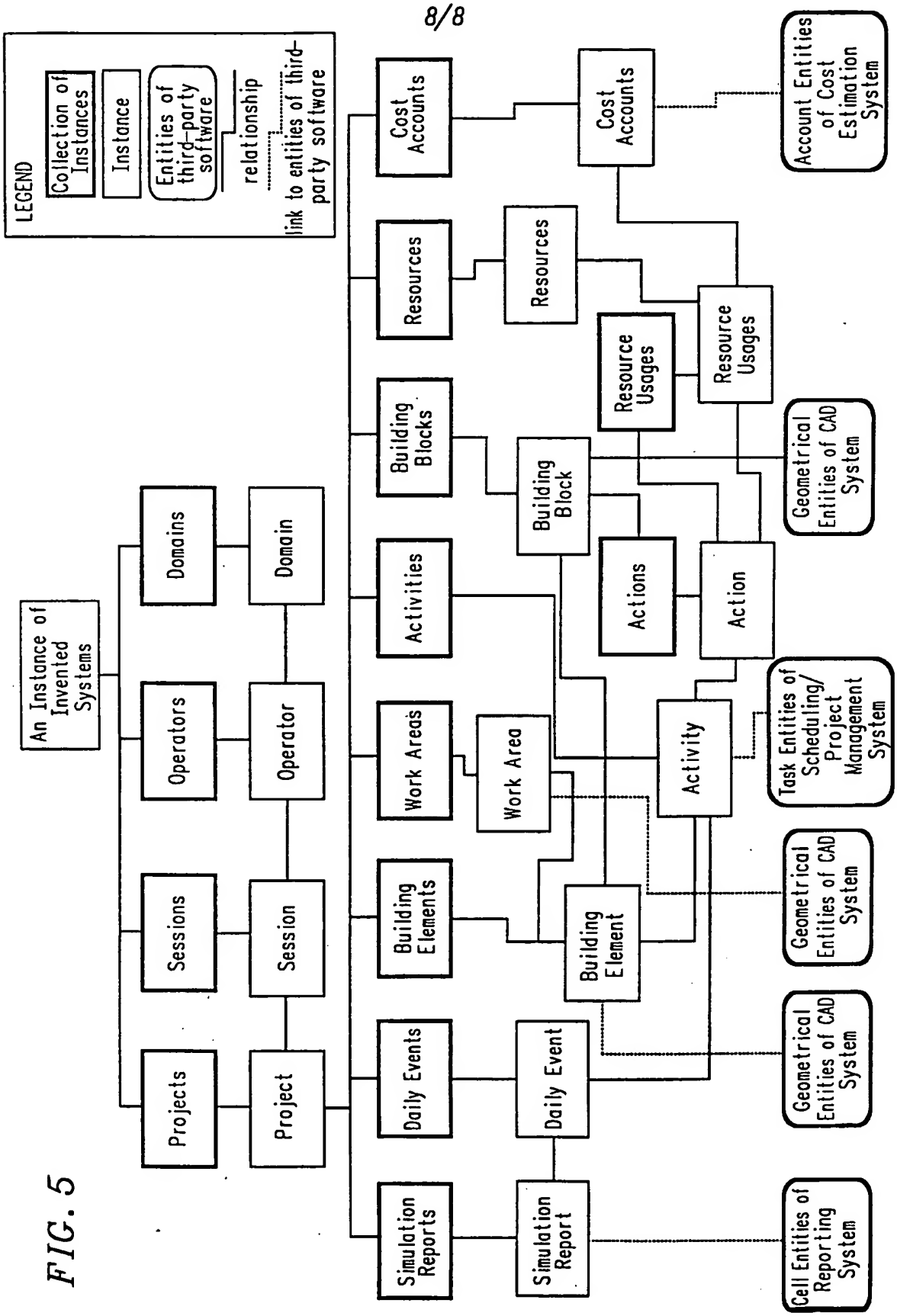

LEGEND
Collection of Instances
Instance
Entities of third-party software
relationship
link to entities of third-party software
An Instance of Invented Systems
Projects
Sessions
Operators
Domains
Project
Session
Operator
Domain
Simulation Reports
Daily Events
Building Elements
Work Areas
Activities
Building Blocks
Resources
Cost Accounts
Simulation Report
Daily Event
Building Element
Work Area
Activity
Building Block
Resources
Cost Accounts
Resource Usages
Actions
Action
Resource Usages
Cell Entities of Reporting System
Geometrical Entities of CAD System
Geometrical Entities of CAD System
Geometrical Entities of CAD System
Task Entities of Scheduling/ Project Management System
Geometrical Entities of CAD System
Account Entities of Cost Estimation System

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/14072

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 17/30

US CL :707/1, 100, 501, 513, 901, 906.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/1, 100, 501, 513, 901, 906.

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,793,966 A (AMSTEIN et al) 11 August 1998, col. 5, lines 1-15, col. 11, lines 48-67, col. 12, lines 1-18 and lines 50-65, col. 13, lines 51-67, col. 14, lines 1-16, col. 15, lines 4-27, col. 16, lines 13-42, col. 18, lines 30-48, col. 19, lines 3-67, col. 20, lines 1-42, col. 23, lines 2-45, col. 24, lines 5-60, and col. 26, lines 9-34.	1-22

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
26 JULY 2000	14 AUG 2000

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	KIM VU James R. Matthews
Facsimile No. (703) 305-3230	Telephone No. (703) 305-4393

Form PCT/ISA/210 (second sheet) (July 1998)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/14072

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,890,171 A (BLUMER et al) 30 March 1999, col. 1, lines 18-25 and 36-67, col. 2, 1-10 and lines 42-67, col. 3, lines 1-65, col. 4, lines 50-67, col. 5, lines 1-4, col. 6, lines 22-43 and lines 58-67, col. 7, lines 1-55, col. 8, lines 30-67, col. 9, lines 1-60, col. 11, lines 17-53, col. 12, lines 7-29 and lines 35-67, col. 13, lines 11-27 and lines 61-67, col. 14, lines 13-32 and lines 36-67, and col. 15, lines 7-20.	1-22

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/14072

B. FIELDS SEARCHED

Electronic data bases consulted (Name of data base and where practicable terms used):

WEST

Search terms: computer aided design or cad, data processing, parsing, target object, html, xml, relational table, database, internet, network.